AD-A056 793 NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA F/G 9/2
PSEUDORANDOM NUMBER GENERATOR. PROGRAM-CONTROLLED SOURCE OF THR--ETC(U)
JUN 78 W G LAFOND

UNCLASSIFIED NOSC/TD-163 NL

| OF |

AD
A056793

END
DATE
FILMED
9-78
DDC

LEVEL II

(12)

AD A056793

NOSC

NOSC TD 163

NOSC TD 163

Technical Document 163

PSEUDORANDOM NUMBER GENERATOR

Program-controlled source of three 15-bit random-number words per microsecond for AP-120B array processors

WG La Fond

1 June 1978

Prepared for
ALWT PROJECT OFFICE
NAVAL SEA SYSTEMS COMMAND (PMS 406)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

NAVAL OCEAN SYSTEMS CENTER
SAN DIEGO, CALIFORNIA 92152

NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA 92152

AN ACTIVITY OF THE NAVAL MATERIAL COMMAND

RR GAVAZZI, CAPT, USN
Commander

HL BLOOD
Technical Director

ADMINISTRATIVE INFORMATION

Work was performed under Program Element 63610N, Project S0199-AS, Task Area S0199103 (NOSC 060-735120-137), by members of the Realtime Simulation Division. This document covers work from March to April 1978 and was approved for publication 1 June 1978.

The author expresses his appreciation to JD Elliott and JA Mayr for their many helpful ideas.

Released by
H Mori, Head
Realtime Simulation
Division

Under authority of
LZ Maudlin, Head
Computer Sciences and
Simulation Department

METRIC EQUIVALENTS

To convert from	to	Multiply by
inches	metres (m)	2.54×10^{-2}

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NOSC Technical Document 163 (TD 163)	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	14 NOSC/TD-163
4. TITLE (and Subtitle): 6 PSEUDORANDOM NUMBER GENERATOR. Program-controlled source of three 15-bit random-number words per microsecond for AP-12ØB array processors.	5. TYPE OF REPORT & PERIOD COVERED: 9 ~~Documentary~~ Technical document March-April 1978.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): 10 W.G. La Fond	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Ocean Systems Center, San Diego, California 92152	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 63610N, S0199-AS, S0199103 (NOSC 060-735120-137)
11. CONTROLLING OFFICE NAME AND ADDRESS: ALWT Project Office, Naval Sea Systems Command (PMS 406), Washington DC 20362	12. REPORT DATE: 11 1 June 1978
	13. NUMBER OF PAGES: 19 12 22p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 16 SØ199AS 17 SØ1991Ø3	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Pseudo random systems
Probability density functions
Shift registers

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The objective of this project was to provide AP-120B array processors with a program-controlled source of 15-bit random-number words at the rate of three per microsecond. A simple TTL circuit was implemented to do this. The implementation and testing of a fast algorithm to generate uniformly distributed random numbers is described. The same shift-register method can be expanded to include a fast RAM memory that will provide random numbers of any given distribution without slowing the controlling array-processor program.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

393 159 ach

OBJECTIVE

Provide the AP-120B array processors with a program-controlled source of 15-bit random-number words at the rate of three per microsecond.

RESULTS

1. Installation of a random-number generator on the back panel of each of five AP-120B array processor computers has been successfully completed. Installation on the four remaining array processors will be completed after acceptance of the hardware.

2. A fast source of n-bit pseudorandom-number words can be implemented with n shift registers operating in parallel, each register supplying one bit to the random number word.

3. The system installed generates numbers with uniform distribution. A memory look-up table can be added for generating numbers of any distribution, without loss of speed.

4. The circuit can be implemented with standard TTL integrated circuits.

5. The programs listed in this report, used to debug the circuits (COMPARE2) and to check for digital noise (COMPARE1 and TIMTEST), showed that the system was sufficiently noisefree and operationally consistent. In one check, the five generators were run continuously, at high speed, for 1 hour. Each generated the same number of random-number words (about 10^{10}), and all stopped on the same words.

RECOMMENDATION

To make the random-number generator more versatile, use the easily accessed word-count register in the array processor for the following purposes:

1. To load the initial conditions into the 15 shift registers. This would allow different sequences of the given distribution.

2. To load a RAM memory look-up table with numbers of a given distribution. This would allow sequences of any distribution to be generated.

CONTENTS

INTRODUCTION

A hardware implementation of the shift-register pseudorandom-number generator described in NOSC TN 345* was attached to the AP-120B computer array processors (AP) in the real-time torpedo simulation system. Installation of a random-number generator on the back panel of each of five AP computers has been successfully completed. Installation on the four remaining array processors will be completed after acceptance of the hardware. This document describes the circuit design of the number generator, its computer interface, the diagnostic software written to test it, and the expansion of this generator to a system that generates numbers of an arbitrary distribution.

CIRCUIT THEORY

The circuit comprises 15 shift registers of 16 bits each. All 15 registers shift simultaneously (in parallel) to generate a new random-number word. Bit 0 of each register holds the exclusive OR of bits 7 and 15. The 15-bit random-number word is composed of bit 1 of each of the 15 shift registers (figure 1). A new number is generated with only one shift; whereas if but a single register were used, about 10 shifts would be required to insure that the numbers were not correlated with each other, increasing the generation time by about a factor of 10.

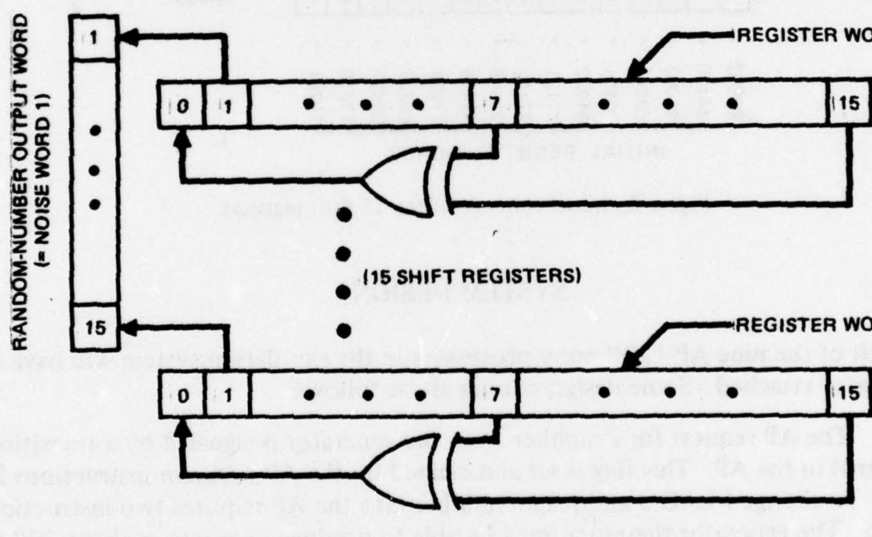

Figure 1. Random-number generation from 15 parallel shift registers.

Each register word, bits 1 thru 15, repeats only every 2^{15} shifts (a maximal-length sequence). It was observed that the random-number word sequence generated by the 15 parallel registers is also of maximal length when the initial conditions (noise words) given in

* NOSC TN 345, A Fast Pseudo-Random Number Generator, by K Lawrence, 1 February 1978. NOSC TNs are informal documents intended chiefly for internal use

TN 345 are used. These 15 initial noise words were generated from a single resister by loading bits 1 through 15 with the sequence 62732 (octal) and forming another sequence by sampling bit 1 every 13 iterations. The initial noise words, listed in figure 2, are successive 15-bit bytes of this sequence.*

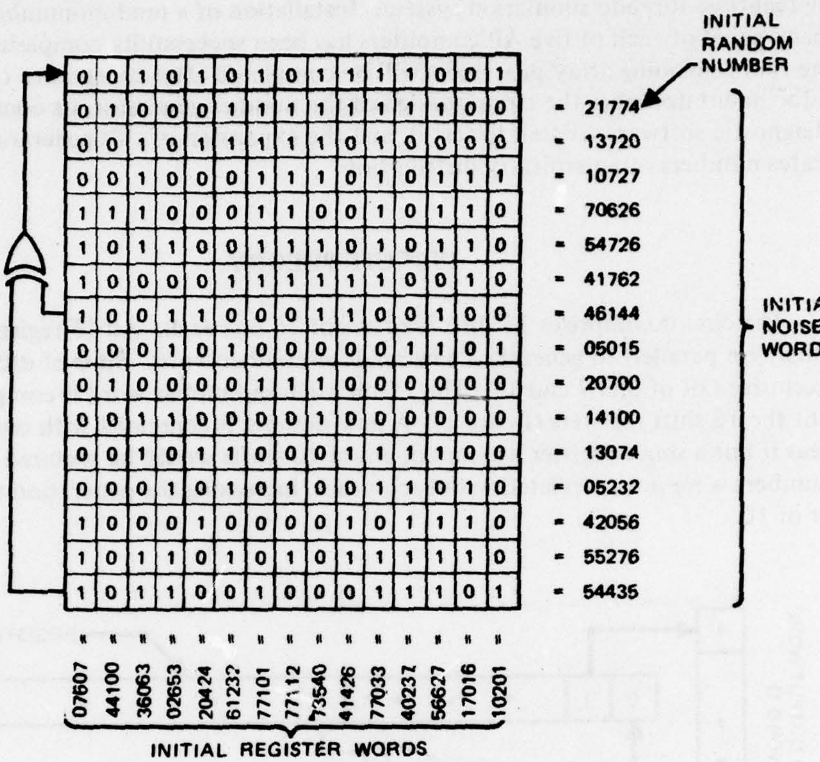

Figure 2. Initial contents of the 15 shift registers.

SYSTEM DESIGN

Each of the nine AP-120B array processors in the simulation system will have its own noise generator attached. Some design criteria are as follows:

1. The AP request for a number from the generator is signaled by a transition of the FLAG 3 signal in the AP. This flag is set and cleared by the AP program instructions SFL3 and CFL3. To change FLAG 3 and read a number into the AP requires two instructions of 167 ns each. The generator therefore must be able to produce numbers in about 200 ns to allow for transmission and AP memory setup time.

2. The FLAG 3 signal can be used together with the AP RUN signal to control resetting of the number sequence. There are two modes of reset:

   a. The generator resets when the RUN signal goes low – ie when the AP starts running – independently of FLAG 3: RESET = $\overline{\text{RUN}}$.

---

* From conversation with K Lawrence, NOSC

b. The generator resets when RUN goes low if FLAG 3 is initially set: RESET = $\overline{\text{RUN}}$ · FLAG 3. With this mode, the number generator can be programmed to start on any of the 32767 numbers in the sequence.

The desired reset mode is selected by a switch on the AP. (Switch 2 open = mode a; closed = mode b.)

3. The port into the AP is the 16-bit HMA2 register. The LSB of HMA2 is always 0, leaving the 15 MSB for the random-number bits. The noise generator is the only source of data at this port. (AP device address = 11.)

CIRCUIT DESIGN

The block diagram, figure 3, shows the interface with the array processor. The shift-register circuit (figure 4) uses the 8-bit parallel-load TTL chip 74198. Two chips are used to make bits 1 through 15 of the register word. Register bit 0 is the output of the exclusive OR chip 7486. There are 15 register circuits, each supplying one bit to the 15-bit random-number word.

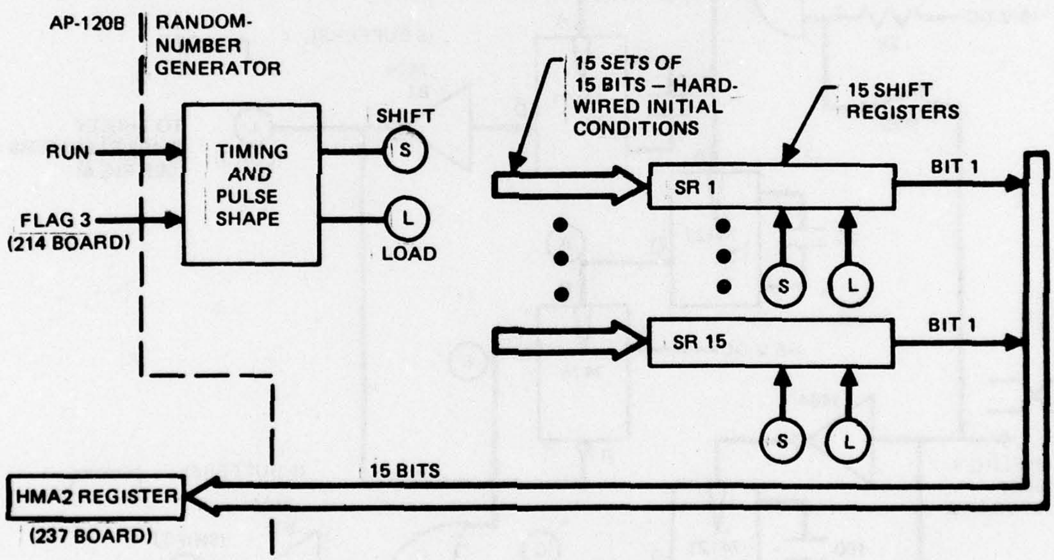

Figure 3. Random-number generator – uniform distribution and fixed initial conditions.

Two inputs to each of the 30 shift-register chips control shifting, (S), and loading or resetting, (L). Resetting, by parallel loading of the hard-wired values at the register inputs, occurs if the L signal is high when S rises.

The *timing* and pulse shaping circuit (figure 5) uses 74121 one-shots for timing. T2 delays the clock pulse with respect to the rise of the load pulse, to satisfy the shift-register setup requirement. The timing is shown in figure 6. The RUN signal is low for the duration of the AP program and FLAG 3 is constant during transitions of RUN.

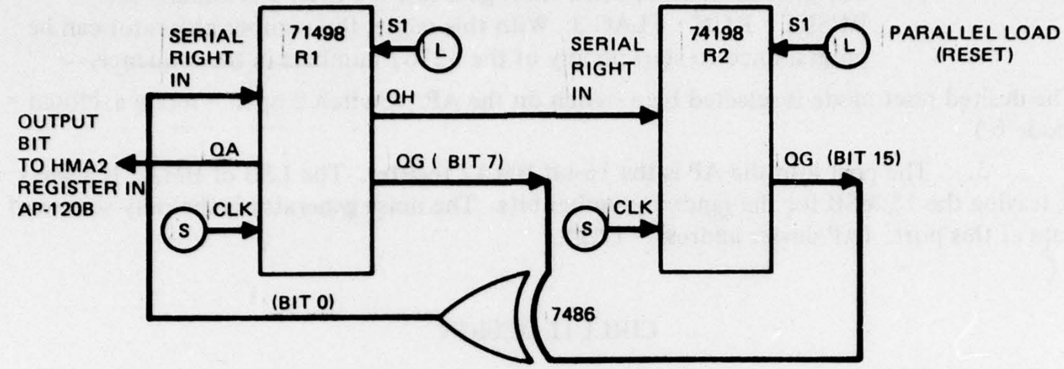

Figure 4. Shift register circuit (1 of 15).

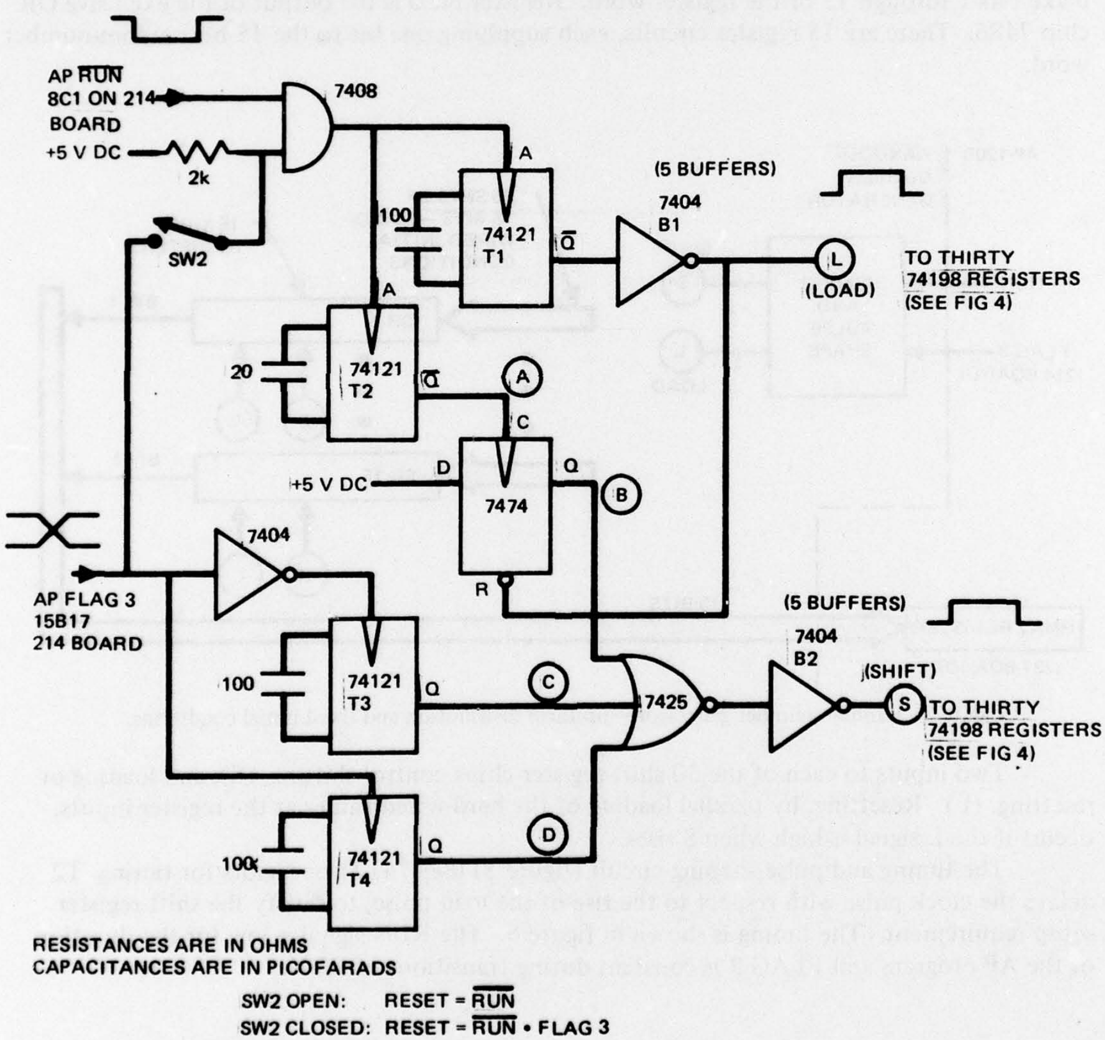

Figure 5. Timing and pulse shape circuit.

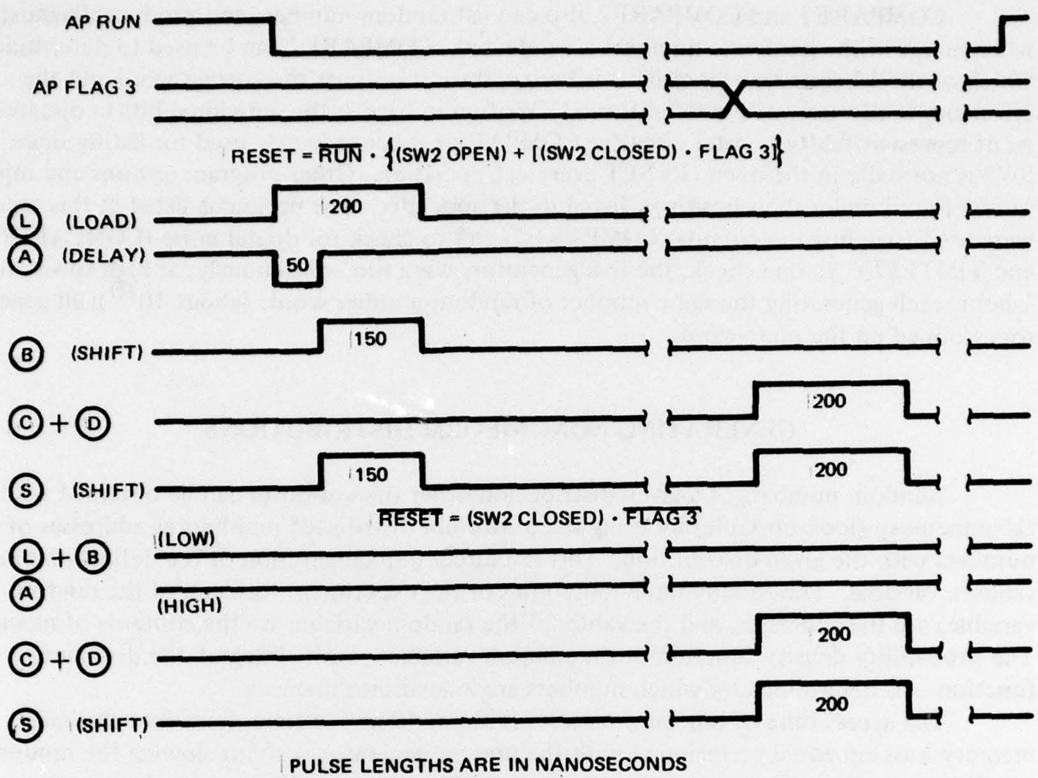

Figure 6. Timing.

DESCRIPTION OF TEST PROGRAMS

Three FORTRAN programs were written to test the random-number hardware. The programs are entered into the host computer (UNIVAC 1110); these call FORTRAN subroutines which in turn enter machine code programs into the AP. The main programs are as follows:

I. COMPARE2 (compare to), which compares an arbitrary length sequence of AP (hardware)-generated numbers to software-generated numbers from the subroutine UFORM2.

II. TIMTEST (time-test), which starts the number generator in a continuous run for a given number of seconds, minutes, or hours. Several APs can run simultaneously with this program; at the end of the run, the APs can be checked against each other to see if the final random number word in each is the same.

III. COMPARE1, which runs the generator over the same sequence a given number of times to check for repeatability.

To generate a continuous sequence of arbitrary length in COMPARE2, SW2 must be closed so that the generator will not reset on subsequent calls to the AP from the host computer. For COMPARE1, the switch must be open so that the generator does reset. TIMRUN will operate with either switch setting.

COMPARE1 and COMPARE2 also can list random numbers and produce diagnostics when errors in the hardware numbers are detected. COMPARE2 can be used to determine which of the 15 shift register circuits is faulty, since it outputs the correct word and the erroneous hardware word in octal format. Written in base 2, the individual bits in disagreement represent faulty register circuits. COMPARE1 is conveniently used for listing since SW2 is normally in the open (RESET every call) position. Other program options and inputs can be found under their headings, listed in the appendix. The programs listed in this report were used to debug the circuits (COMPARE2) and to check for digital noise (COMPARE1 and TIMTEST). In one check, the five generators were run continuously, at high speed, for 1 hour, each generating the same number of random-number words (about 10^{10}); all generators stopped on the same word.

GENERATING NONUNIFORM DISTRIBUTIONS

Random numbers of a given distribution other than uniform can be obtained with a table memory (look-up table) by using the uniformly distributed numbers as addresses of the numbers with the given distribution. This is a direct implementation of the definition of a random variable. The equally likely outcomes of the experiment (domain of the random variable) are the addresses, and the values of the random variable are the contents of memory. The probability density function of the random variable – or its integral, the distribution function – is determined by which numbers are loaded into memory.

The access time of bipolar memories is about 40 ns – a time short enough that a memory look-up could be included with the present generator without slowing the random-number generation. The maximum output rate will still be one random-number word every two instruction cycles (334 ns). The memory can be PROM or RAM. If RAM is used, different distributions are programatically available. A convenient register is included on the AP 237 board through which the RAM can be loaded: the word count (WC) register, adjacent to the HMA2 register. The shift-register initial words can also be loaded via the WC register, to make available different sequences of uniformly distributed numbers.

The numbers to be loaded originate in the 1110 host computer. They are read into the AP memory (MD), then passed one at a time through the WC register to the RAM or to the shift-register inputs. Figure 7 is a block diagram of a design that allows the loading of both a RAM and the initial register words and provides two programmably selectable outputs – either the uniformly distributed words from the shift registers (the RAM addresses) or the nonuniform words from RAM. Although uniform distributions are available from RAM, sequences from RAM are limited by its size (about 4k words).

The WC register presents a control word before each data word. Control words are distinguished by those which have 1 as the most significant bit. The other bits in a control word are used to indicate where the following word of data should be put, to enable busses, and to set the output select latch. A data word is loaded into a shift register or RAM after a delay initiated by the falling edge of WC bit 0. Figure 8 shows the bit assignments of the WC register.

The full circuit shown in figure 7 requires three circuit boards, each about the same size as the present one (6 by 10 inches). They are a shift register board, a RAM board, and a control board.

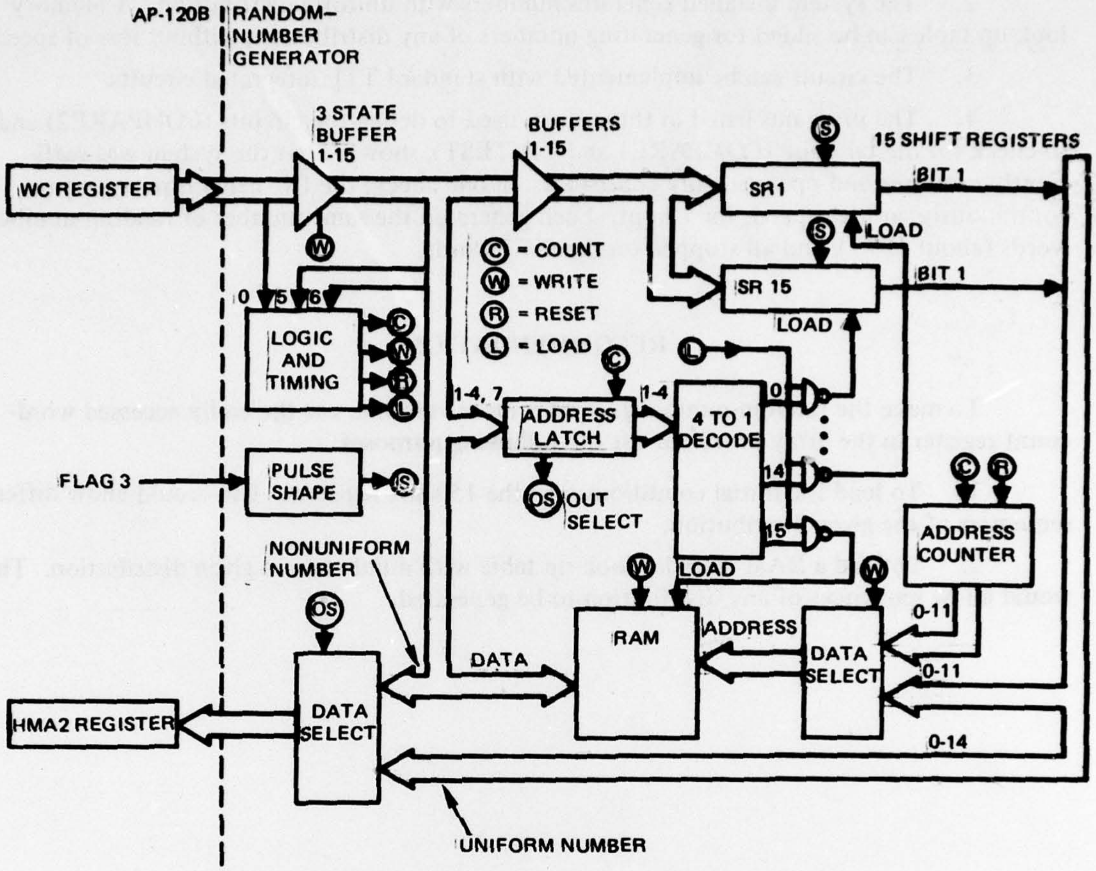

Figure 7. Random-number generator – arbitrary distribution and initial conditions.

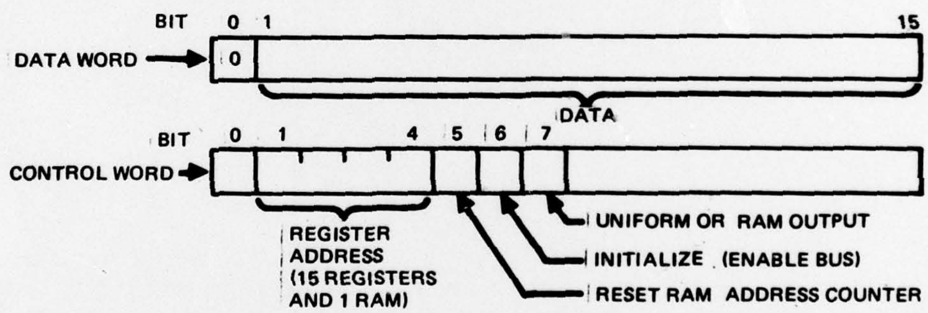

Figure 8. Bit assignment in AP WC register.

CONCLUSIONS

1. A fast source of n-bit pseudorandom-number words can be implemented with n shift registers operating in parallel, each register supplying one bit to the random number word.

2. The system installed generates numbers with uniform distribution. A memory look-up table can be added for generating numbers of any distribution, without loss of speed.

3. The circuit can be implemented with standard TTL integrated circuits.

4. The programs listed in this report, used to debug the circuits (COMPARE2) and to check for digital noise (COMPARE1 and TIMTEST), showed that the system was sufficiently noisefree and operationally consistent. In one check, the five generators were run continuously, at high speed, for 1 hour. Each generated the same number of random-number words (about 10^{10}), and all stopped on the same words.

RECOMMENDATION

To make the random-number generator more versatile, use the easily accessed word-count register in the array processor for the following purposes:

1. To load the initial conditions into the 15 shift registers. This would allow different sequences of the given distribution.

2. To load a RAM memory look-up table with numbers of a given distribution. This would allow sequences of any distribution to be generated.

APPENDIX
TEST PROGRAMS

```
C         PROGRAM TIMTEST
C         TO TEST THE RANDOM NUMBER GENERATOR ATTACHED TO THE AP120B
C         ARRAY PROCESSOR. RUNS THE NUMBER GENERATOR IN NESTED LOOPS,
C         EACH LOOP READING 2 WORDS INTO THE AP MEMORY (MD 0). TO READ
C         ONE WORD REQUIRES 2 INSTRUCTIONS=334 NS. THE TOTAL RUN TIME IS
C         CLOSE TO THE PRODUCT OF THE THREE LOOP COUNTERS (N215, N230, N245)
C         AND 334 NS.
C         FOR SHORT RUNS (LESS THAN 1 SEC.) THE AP WILL WRITE THE FINAL WORD TO
C         THE 1110 HOST COMPUTER. FOR LONGER RUNS THIS PROGRAM MUST BE TERMINATED
C         AND THE FINAL WORD READ FROM THE AP (FROM MD 0)
C         WITH THE SYSTEM PROGRAM BUG.1 OR BUG.2. THE TERMINAL INPUTS ARE:
C           @XQT BUG.1
C           (ARRAY PROCESSOR NO., 0 - 9, 11)
C           E    (EXAMINE)
C           MD   (MEMORY)
C           0    (ADDRESS ZERO)
C           X    (TO TERMINATE PROGRAM)
C
C         INPUTS
C
C         N215= PRIMARY LOOP COUNT (32766 MAX)
C         N230= 2-ARY      "       (32767 MAX)
C         N245= 3-ARY      "           "
C
C         TO RUN 1 SEC SET  N215=32766
C                           N230=92
C                           N245=1
C
C         TO RUN 1 MIN SET  N215=32766
C                           N230=  92
C                           N245=60
C
C         TO RUN 1 HOUR SET N215=32766
C                           N230=92
C                           N245=3600
C
C         ONE HOUR REPRESENTS 32766*92*3600= 10**10 SHIFTS OF THE NOISE REGISTERS.
C         BILL LAFOND APR 78
C
        1 FORMAT(I5)
        2 FORMAT(T5,O6)
          CALL APCLR
          READ(5,1) N215,N230,N245
C         SET FLAG 3
          CALL TIMRUN(2,0,0)
          CALL APWR
C         START TIMED RUN
          CALL TIMRUN(N215,N230,N245)
          CALL APWR
          CALL APGET(NFINAL,0,1,0)
          CALL APWD
          WRITE(6,2) NFINAL
          STOP
          END
```

```
      $TITLE HARDWARE RANDOM NUMBER TIME TEST PROGRAM TIMRUN
      $ENTRY TIMRUN,3
      "  RUNS RANDOM NO. GEN IN NESTED LOOPS SO THAT IT WILL RUN A PREDETRMINED
      "  LENGTH OF TIME, E.G., HOURS, TO TEST IF ALL OF THE ARRAY PROCESSORS
      "  ARE RUNNING CONSISTANTLY AND THE SAME.
      "  SPAD0= COUNT UP TO 2**15 - 2 ITERATIONS = 11 MS. MAX.
      "  SPAD1=  "          2**30 - 1      "      = 6 MINUTES IF SP0 AND SP1=MAX
```

```
      "  SPAD2=  "            2**45 - 1        "        = 1 HOUR IF =10 AND OTHERS MAX.
      "  TOTAL TIME= THE PRODUCT 334 NS. *(SPAD0)*(SPAD1)*(SPAD2)
      "  BILL LAFOND APR 78
      "
      TIMRUN: NOP
      DB=ZERO; LDAPS
      DB=0; LDSPI 15
      MOV 15,15; SETMA; DB=ZERO; MI<DB
      DEC 0; INCMA
      MOV 0,3
      MOV 1,4
      LDDA; DB=11
      LL: IN; DB=INBS; MI<DB; WRTLMN; DECMA
      DEC 0; CFL3; INCMA
      IN; DB=INBS; MI<DB; WRTLMN; DECMA
      DEC 0; BGT LL; SFL3; INCMA
      DEC 1
      BGT LL; MOV 3,0
      DEC 2
      BGT LL; MOV 4,1
      RETURN
      SEND
```

```
C        PROGRAM COMPARE1
C        TO TEST THE RANDOM NUMBER GENERATOR ATTACHED TO THE AP 120B
C        ARRAY PROCESSOR. READS NS SEQUENCES OF RANDOM NO.'S, EACH
C        SEQUENCE NR WORDS LONG. SWITCH 2 ON THE RANDOM NO. GENERATOR SHOULD
C        BE OPEN SO THAT THE NUMBER GENERATOR RESETS ON EVERY CALL TO THE AP.
C        THE PROGRAM
C
C          1.  COMPARES EACH SET TO THE PROCEEDING BY COMPARING TEST SUMS
C              COMPUTED IN THE AP. IF THE SUMS ARE DIFFERENT THE WORDS ARE READ
C              FROM THE AP.
C
C          2.  CHECKS THAT SUCCESSIVE WORDS ARE DIFFERENT.
C
C
C
C
C        INPUTS
C
C          NS=NO. SETS TO BE COMPARED (L.E. 99999)
C          NR=NO. RANDOM NO.'S IN EACH SET (L.E. 2000)
C          NDIAG=1 TO LIST RANDOM NO.'S AND DIAGNOSTICS. TO LIST RANDOM NO.'S
C               SET NS=1, NDIAG=1, NR=LENGTH OF SEQUENCE DESIRED
C          NPASS=1 TO MARK END OF EACH COMPARISON
C
C
C        BILL LAFOND APR 78
         INTEGER R(2000), W(2000)
C
    1    FORMAT(I5)
    2    FORMAT(10(5X,O6))
    3    FORMAT(/T5,'SET NO. 1')
    4    FORMAT(T5,'DIFFERENCE FOUND IN RANDOM NO. ',I5,
     C   ' OF SET NO. ',I5,)
    5    FORMAT(T5,'PASS ',I5,' FINISHED'/)
    6    FORMAT(T8,'PRESENT WORD= ',O6,'  LAST WORD= ',O6,)
    7    FORMAT(T5,'******  WORD ',I6,' IS SAME AS LAST ONE - ',O6,
     C   ' IN PASS ',I6,)
    8    FORMAT(T5,'16 LSB OF SUM = ',O6)
   10    FORMAT(/T5,'END TEST')
C
         CALL APCLR
```

```
      READ(5,1) NS,NR,NDIAG,NPASS
C
      CALL RAND1(C,1,NR,4096)
      CALL APWR
      CALL APGET(LSUM1,4096,1,0)
      CALL APWD
      IF(NDIAG.NE.0) WRITE(6,8) LSUM1
C
C     WRITE FIRST SET OF RANDOM NO.'S IN OCTAL
C
      CALL APGET(R,C,NR,0)
      CALL APWD
      J1=1
      DO 15 I=2,NR
      IF(R(I).EQ.R(I-1)) WRITE(6,7) I,R(I),J1
15    CONTINUE
      IF(NDIAG.NE.0) WRITE(6,3)
      IF(NDIAG.NE.0) WRITE(6,2) (R(I),I=1,NR)
      IF(NPASS.NE.0) WRITE(6,5) J1
C
      IF(NS.EQ.1) GO TO 999
      DO 100 K1=2,NS
      CALL RAND1(C,1,NR,4096)
      CALL APWR
      CALL APGET(LSUM2,4096,1,0)
      CALL APWD
      IF(NDIAG.NE.0) WRITE(6,8) LSUM2
      LDIF=LSUM1-LSUM2
      IF(LDIF.EQ.0) GO TO 99
C     RETRIEVE PRESENT SET THAT DOESN'T AGREE WITH LAST ONE
C
      CALL APGET(W,C,NR,0)
      CALL APWD
      IF(W(1).NE.R(1)) WRITE(6,4) J1,K1
      IF(W(1).NE.R(1)) WRITE(6,6) W(1),R(1)
      DO 20 I=2,NR
      IF(W(I).EQ.W(I-1)) WRITE(6,7) I,W(I),K1
      IF(W(I).EQ.W(I-1)) GO TO 21
      IF(W(I).EQ.R(I)) GO TO 20
      WRITE(6,4) I,K1
      WRITE(6,6) W(I),R(I)
      GO TO 21
20    CONTINUE
21    IF(NDIAG.NE.0) WRITE(6,2) (W(I),I=1,NR)
      DO 22 I=1,NR
22    R(I)=W(I)
99    IF(NPASS.NE.0) WRITE(6,5) K1
      LSUM1=LSUM2
100   CONTINUE
999   CONTINUE
      WRITE(6,10)
      STOP
      END
```

```
           $TITLE HARDWARE RANDOM NUMBER TEST PROGRAM RAND1
      $ENTRY RAND1,4
      $EXT VCLR
      " BILL LAFOND APRIL 76
      " FOR NOSC SPECIAL MODIFICATION TO ATTACH HARDWARE RANDOM NO.
      " GENERATOR TO THE AP 120B.
      " LOADS THE FIRST (SPAD2) LOCATIONS IN MD MEMORY WITH RANDOM NO.'S
      " AND CALLS SUBROUTINE SVEI TO SUM THE NO.'S AND PLACES THE SUM
      " IN LOCATION (SPAD3) IN MD.
```

```
" RAND1 IS CALLED BY PROGRAM COMPARE1.
" SWITCH 2 ON THE RANDOM NO. GEN. SHOULD BE OPEN.
"
"  SPAD0= BASE ADDRESS
"  SPAD1= INCREMENT
"  SPAD2= NO. OF RANDOM NO.'S
"  SPAD3= MEMORY LOCATION OF TEST SUM
RAND1: NOP
MOV 0,7
MOV 1,10
MOV 2,11
MOV 3,12
DB=ZERO; LDAPS
MOV 0,c
JSR VCLR
MOV 6,0
CFL3        "INITIALIZE FLAG 3
NOP
LDDA; DB=11
LDSPI 6; DB=177777
MOV 6,c; SETMA
MOV 2,6  "SP6 IS COUNTER REG
DEC 6
LL: IN; DB=INBS; MI<DB; INCMA; WRTLMN
SFL3; DEC 6
IN; DB=INBS; MI<DB; INCMA; WRTLMN
CFL3; DEC 6;  BGT LL
NOP
NOP
JSR SVEI
MOV 7,0
MOV 10,1
MOV 11,2
MOV 12,3
RETURN
SVEI: NOP  "ADDS CONTENTS OF MD - SP0= BASE ADD, SP1= INCREMENT
MOV 0,0; SETMA   "...SP2=NO. OF WORDS, SP3= DEST. ADD
LDSPI 6; DB=0
MOV 2,2; DB=ZERO; DPY(0)<DB  "FLOAT. 0 TO DPY(0)
LOOP: LDSPI 5; DB=MD; BEQ LOOPEND
ADD 1,0; SETMA
ADD 5,6
DEC 2; BR LOOP
LOOPEND: NOP
MOV 6,6; DB=SPFN; DPX(0)<DB
DB=DPX(0); DPY(0)<DB; WRTLMN
MOV 3,3; SETMA; MI<DPY(0)
RETURN
SEND
```

```
C       PROGRAM COMPARE1
C       TO TEST THE RANDOM NUMBER GENERATOR ATTACHED TO THE AP 120B
C       ARRAY PROCESSOR. THE PROGRAM READS A CONTINUOUS SEQUENCE OF
C       RANDOM NUMBERS, ESSENTIALLY UNLIMITED IN LENGTH, BY READING NS
C       SETS  OF NR WORDS (RANDOM NO.'S) EACH.
C
C       WITH THIS SEQUENCE, THE PROGRAM
C
C          1.  SEARCHES FOR THE REPETITION OF A WORD OF GIVEN SEQUENCE NO. IN SET
C              1, AND OUTPUTS THE WORD AND THE SEQUENCE NO. OF THE REPEATED WORD.
C              ANY WORD REPEATS EVERY 2**15 ITERATIONS.
C
C          2.  SEARCHES FOR A WORD AND OUTPUTS ITS SEQUENCE NO.
C
```

```
C          3.  COMPUTES THE RANDOM NO.'S WITH THE SUBROUTINE UFORM2 AND
               COMPARES THE COMPUTED SEQUENCE TO THE HARDWARE (AP)-GENERATED
               WORDS.
C
C       TO COMPARE THE SAME SEQUENCE WITH ITSELF N TIMES SEE THE PROGRAM
C         COMPARE1. BILL LAFOND APR 78
C
C       INPUTS
C
C         NS=NO. SETS TO BE GENERATED (L.E. 99999)
C         NR=NO. RANDOM NO.'S IN EACH SET (L.E. 2000)
C         NDIAG = 1 TO LIST WORDS, 0 TO NOT LIST
C         NUDIAG =1 TO PRINT WORDS FROM SUB. UFORM2
C         NPASS = 1 TO MARK ENDS OF LOOPS
C         NCHK = SEQ. NO. OF WORD IN FIRST SET TO BE COMPARED TO ALL OTHERS
C              =0 TO SKIP
C         NCMPR=1 TO COMPARE UFORM2 (SOFTWARE) WORDS TO HARDWARE WORDS
C              =0 TO SKIP
C         NFIND= (OCTAL) WORD TO BE FOUND - OUTPUTS SEQ. AND PASS NO.
C            = 0 TO SKIP
C
        INTEGER R(2000),UR(2000)
C
      1 FORMAT(I5)
      2 FORMAT(10(5X,O6))
      3 FORMAT(/T5,'SET NO. 1')
      4 FORMAT(/T5,'RANDOM NUMBERS - SET ',I5,)
      5 FORMAT(T5,'PASS ',I5,' FINISHED'/)
      7 FORMAT(T5,'WORD ',I6,' IS SAME AS WORD ',I6,' = 'O6)
      8 FORMAT(T5,'?????? SOFTWARE WORD= ',O6,' HARDWARE WORD= ',
     C   O6,'  WORD NO. ',I6)
     10 FORMAT(T5,'END SOFT - HARD COMPARE')
     11 FORMAT(O6)
     12 FORMAT(T5,'SEQUENCE NO. ',I6,' IS RANDOM NO. ',O6)
C
        CALL APCLR
        READ(5,1) NS,NR,NDIAG,NUDIAG,NPASS,NCHK,NCMPR
        READ(5,11) NFIND
C
        CALL RAND2(0,1,2,1)
        CALL APWR
        CALL RAND2(0,1,NR,0)
        CALL APWR
        NR1=NR+1
        CALL APGET(R,0,NR1,0)
        CALL APWD
C
C       WRITE FIRST SET OF RANDOM NO.'S IN OCTAL
C
        J1=1
        IF(NCHK.EQ.0) GO TO 19
        CKWD=R(NCHK)
        DO 15 I=1,NR1
        IF((CKWD.NE.R(I)).OR.(NCHK.EQ.I)) GO TO 15
        WRITE(6,7) I,NCHK,R(I)
     15 CONTINUE
     19 CONTINUE
        IF(NDIAG.NE.0) WRITE(6,4) J1
        IF(NDIAG.NE.0) WRITE(6,2) (R(I),I=1,NR1)
        IF(NCMPR.EQ.0) GO TO 30
        CALL UFORM2(UR,NR1,1,NUDIAG)
        DO 32 I=1,NR1
        IF(UR(I).NE.R(I)) WRITE(6,8) UR(I),R(I),I
     32 CONTINUE
        IF(NPASS.NE.0) WRITE(6,10)
     30 CONTINUE
        IF(NFIND.EQ.0) GO TO 40
```

```
            DO 42 I=1,NR1
            IF(R(I).EQ.NFIND) WRITE(6,12) I,NFIND
   42       CONTINUE
   40       CONTINUE
            IF(NPASS.NE.0) WRITE(6,5) J1
      C
            IF(NS.EQ.1) GO TO 999
            DO 100 K1=2,NS
            K4=NR1+(K1-2)*NR
            CALL RAND2(C,1,NR,0)
            CALL APWR
      C
            CALL APGET(R,U,NR,0)
            CALL APWD
            IF(NCHK.EQ.0) GO TO 21
            DO 17 I=1,NR
            NSEQ=K4+I
            IF(CKWD.EQ.R(I)) WRITE(6,7) NSEQ,NCHK,R(I)
   17       CONTINUE
   21       CONTINUE
            IF(NDIAG.NE.0) WRITE(6,4) K1
            IF(NDIAG.NE.0) WRITE(6,2) (R(I),I=1,NR)
            IF(NCMPR.EQ.0) GO TO 20
            CALL UFORM2(UR,NR,0,NUDIAG)
            DO 22 I=1,NR
            NSEQ=K4+I
            IF(UR(I).NE.R(I)) WRITE(6,8) UR(I),R(I),NSEQ
   22       CONTINUE
            IF(NPASS.NE.0) WRITE(6,10)
   20       CONTINUE
            IF(NFIND.EQ.0) GO TO 50
            DO 52 I=1,NR
            IF(R(I).EQ.NFIND) GO TO 51
   52       CONTINUE
            GO TO 50
   51       NSEQ=K4+I
            WRITE(6,12) NSEQ,NFIND
   50       CONTINUE
   99       IF(NPASS.NE.0) WRITE(6,5) K1
  100       CONTINUE
  999       CONTINUE
            STOP
            END
```

```
            SUBROUTINE UFORM2(R,NR,NRES,NDIAG)
      C
      C     GENERATES UNIFORMLY DISTRIBUTES RANDOM NUMBERS FROM A
      C      15 BIT SHIFT REGISTER (SEE NOSC TN 345 BY K. LAWRENCE)
      C     AND APPENDS A 0 TO THE LSB OF THE NUMBER WORD TO CAUSE
      C      AGREEMENT WITH THE HARDWARE GENERATED WORDS IN THE AP120 B
      C      ARRAY PROCESSOR (WHA BIT 16 IS ALWAYS 0).
      C
      C     OUTPUT
      C
      C     R=ARRAY OF RANDOM NUMBERS (INTEGER OCTAL) DIMENSIONED IN CALLING
      C      PROGRAM.
      C
      C     INPUT
      C
      C     NR= NO OF NUMBERS TO BE GENERATED
      C     NRES=1 TO RESET THE SEQUNCE, 0 TO NOT
      C     NDIAG=1 TO PRINT DIAGNOSTICS
      C
```

```
C       ADAPTED FRCP THE ORIGINAL BY KAHEN LAWRENCE - BILL LAFOND APR 78
C
C
        DIMENSION INUM(15),NUM(15)
        INTEGER R(1)
C       SHIFT SAMPLING BY 13 FOR SEED NOS.
        DATA INUM/021774,013720,010727,070626,054726,
     C    041762,046144,005015,020700,014100,
     C    013074,005232,042056,055276,054435/
C
1       FORMAT(/T5,'UNSHIFTED 15 BIT INITIAL CONDITION WORDS FROM '
     C    ,'UFORM2'/)
2       FORMAT(/T5,'SOFTWARE GENERATED RANDOM NUMBERS'/)
3       FORMAT(10(5X,06))
C
        IF(NRES.EQ.0) GO TO 15
        DO 20 I=1,15
20      NUM(I)=INUM(I)
15      CONTINUE
        IF((NDIAG.EQ.0).OR.(NRES.EQ.0)) GO TO 18
        WRITE(6,1)
        WRITE(6,3) (NUM(I),I=1,15)
C
C
18      DO 100 K1=1,NR
C       MULTIPLY BY TWO TO SHIFT ONE BIT (MMA15=0)
        R(K1)=NUM(1)*2
        I7=NUM(7)
        I15=NUM(15)
        NEW=XOR(I7,I15)
        DO 10 I=15,2,-1
        NUM(I)=NUM(I-1)
10      CONTINUE
        NUM(1)=NEW
100     CONTINUE
        IF(NDIAG.NE.0) WRITE(6,2)
        IF(NDIAG.NE.0) WRITE(6,3) (R(I),I=1,NR)
        RETURN
        END
```

```
        $TITLE HARDWARE RANDOM NUMBER TEST PROGRAM RAND2
    $ENTRY RAND2,4
    $EXT VCLR
    " BILL LAFOND MARCH 78
    " FOR NOSC SPECIAL MODIFICATION TO ATTACH HARDWARE RANDOM NO.
    " GENERATOR TO THE AP 120B.
    " LOADS THE FIRST (SPAD2) LOCATIONS IN MD MEMORY WITH RANDOM NO.'S
    " IF FLAG 3 IS SET WHEN THIS SUBROUTINE IS CALLED THE SEQUENCE
    " IS INITIALIZED. OTHERWIZE THE SEQUENCE CONTINUES FROM THE
    " LAST WORD GENERATED. SWITCH 2 ON THE RANDOM NO. GEN. SHOULD BE CLOSED
    " THE FORTRAN SUBROUTINE RAND2 IS CALLED BY PROGRAM COMPARE2.
    " SPAD0= BASE ADDRESS IN MD MEMORY
    " SPAD1= MEMORY INCREMENT
    " SPAD2= NO. OF RANDOM NO.'S TO BE GENERATED
    " SPAD3= RESET FLAG (SETS FL3) - 1=RESET NEXT CALL, 0= NOT
    "
    RAND2: NOP
    MOV 0,7
    MOV 1,10
    MOV 2,11
    DB=ZERO; LDAPS
    MOV 0,6
```

```
INC 2
JSR VCLR
DEC 2
MOV 6,0
NOP
LDDA; DB=11
LDSPI 6; DB=177777
MOV 6,6; SETMA
MOV 2,6  "SP6 IS COUNTER REG
DEC 6
BFL3 LL1
LL: IN; DB=INBS; MI<DB; INCMA; WRTLMN
SFL3; DEC 6
LL1: IN; DB=INBS; MI<DB; INCMA; WRTLMN
CFL3; DEC 6;  BGT LL
NOP
NOP
MOV 7,0
MOV 10,1
MOV 11,2
MOV 3,3  "CHECK IF RESET NEXT TIME
BEQ NORES
SFL3
NORES: NOP
RETURN
SEND
```